

# TD MOA-ZS

## Technische Dokumentation zu MOA-ZS

### Dokumentinformation

Bezeichnung	Technische Dokumentation zu MOA-ZS
Kurzbezeichnung	TD MOA-ZS
Version	1.1.0
Datum	10.11.2008
Dokumentenklasse	Technische Dokumentation
Dokumentenstadium	Interner Entwurf.
Kurzbeschreibung	Dieses Dokument beschreibt die technischen Details der Applikation MOA-ZS
Autoren	DI Arne Tauber Nikolaus Gradwohl
Arbeitsgruppe	MOA-ZS

# Inhalt

1.	Allgemeines .....	4
1.1	Synchroner Teil .....	4
1.1.1	Übernahme von Zustellstücken .....	4
1.1.2	Vollständigkeitsprüfung.....	5
1.1.3	Umrechnung der bPK .....	5
1.1.4	Prüfen der Adressierbarkeit .....	5
1.2	Asynchroner Teil .....	5
1.2.1	Aufbereiten des Zustellstücks .....	7
1.2.2	Signieren des Zustellstücks .....	7
1.2.3	Erstellung eines S/MIME Containers .....	7
1.2.4	Übergabe an den Zustellserver.....	8
1.2.5	Benachrichtigung des Absenders .....	8
2.	Detaillierte Beschreibung von MOA-ZS.....	9
2.1	Funktionen .....	9
2.1.1	Vollständigkeitsprüfung.....	9
2.1.2	Lastverteilung .....	9
2.1.3	Auswahl des Zustellservers .....	9
2.1.4	Watchdog-Thread.....	10
2.1.5	Signaturerstellung.....	10
2.1.6	Verschlüsselung .....	10
2.2	Schnittstellen.....	10
2.2.1	Applikation an MOA-ZS .....	10
2.2.2	MOA-ZS an Applikation .....	14
2.2.3	Zusekopf.....	14
2.2.4	Stammzahlenregister.....	14
2.2.5	MOA-SS .....	15
2.2.6	Zustellserver .....	15
2.3	Datenorganisation .....	15
2.3.1	Tabelle payload .....	15
2.3.2	Tabelle xmldocument.....	16
2.3.3	Tabelle rcpt.....	16
2.3.4	Tabelle identification .....	16
2.3.5	Tabelle internetaddress .....	16
2.3.6	Tabelle telephone .....	16
2.3.7	Tabelle postaladdress.....	16
2.3.8	Tabelle Status .....	16
2.3.9	Tabelle delivery_quality .....	16
2.3.10	Tabelle zuseserver .....	16

43	2.3.11	Tabelle queue_info .....	16
44	3.	Referenzen .....	17
45	4.	Historie.....	18
46	5.	Anhang .....	19
47	5.1	Begriffe und Abkürzungen .....	19
48	5.2	Tabellenverzeichnis.....	19
49	5.3	Abbildungsverzeichnis.....	19
50	5.4	Konfigurationsdatei moazs_config.xml .....	20
51			

# 1. Allgemeines

MOA-ZS verarbeitet Zustellstücke in zwei Phasen. Zunächst werden die Zustellstücke in einer synchronen Phase angenommen und überprüft, dann werden sie in einer asynchronen Phase, ohne Verbindung zur Absenderapplikation aufbereitet und an den Zustellserver übergeben.

## 1.1 Synchroner Teil

Abbildung 1, „Synchroner Teil der Requestverarbeitung“ zeigt den Ablauf der synchronen Phase, der im Folgenden detailliert beschrieben wird.

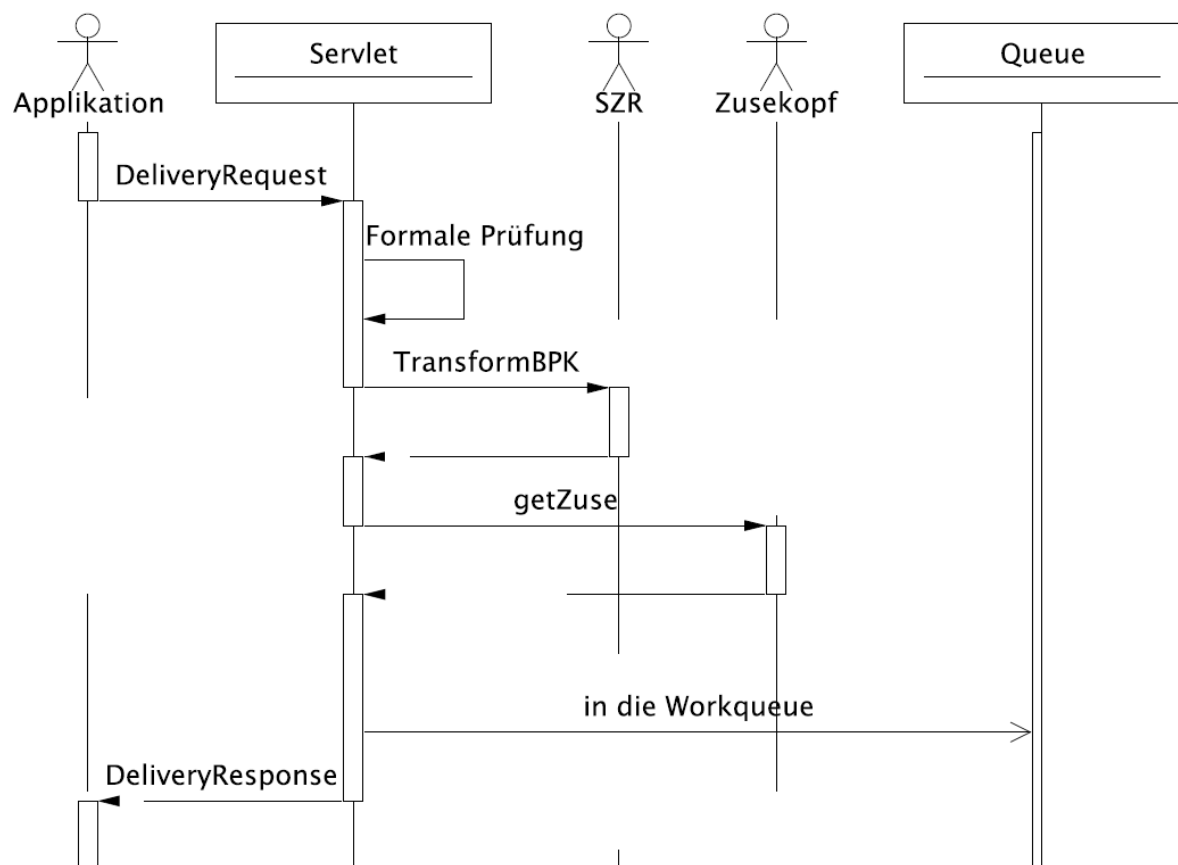


Abbildung 1: Synchroner Teil der Requestverarbeitung

### 1.1.1 Übernahme von Zustellstücken

Zustellstücke werden von der Absenderapplikation an MOA-ZS mit einem SOAP-Request übergeben. Dabei wird eine Nachricht im `DeliveryRequest` Format erstellt. XML-Dokumente, die mit MOA-ZS zugestellt werden sollen, werden direkt in den Request als Kindsknoten eingefügt. Sollen Binärdateien verschickt werden, gibt es drei Möglichkeiten der Übergabe an MOA-ZS:

- Übergabe als `BinaryDocument` im Base64-Encoding
- Übergabe als SwA-Attachment
- Übergabe als Callback-URI

Eine genaue Beschreibung der Schnittstelle mit Beispielen befindet sich im Kapitel 2.2 Schnittstellen.

## 1.1.2 Vollständigkeitsprüfung

Ein Servlet übernimmt den SOAP-Request als `DOM` entgegen und überprüft die im Schema bzw. in der Spezifikation geforderten Pflichtfelder. Wenn eine der Pflichtfeldprüfungen fehlschlägt, wird eine Fehlermeldung generiert und an die Absenderapplikation zurückgeschickt.

## 1.1.3 Umrechnung der bPK

Falls die Vollständigkeitsprüfung erfolgreich ist, wird versucht, einen Zustellserver für den Empfänger zu finden. Dabei wird zuerst geprüft, ob eine bPK mitgeschickt worden ist. Falls es sich nicht um eine (verschlüsselte) bPK des Bereiches "Zustellung" handelt, wird das Stammzahlregister dazu verwendet, um mittels dessen Funktion `TransformBPK` die übergebene bPK in eine verschlüsselte bPK des Bereichs „Zustellung“ umzurechnen.

## 1.1.4 Prüfen der Adressierbarkeit

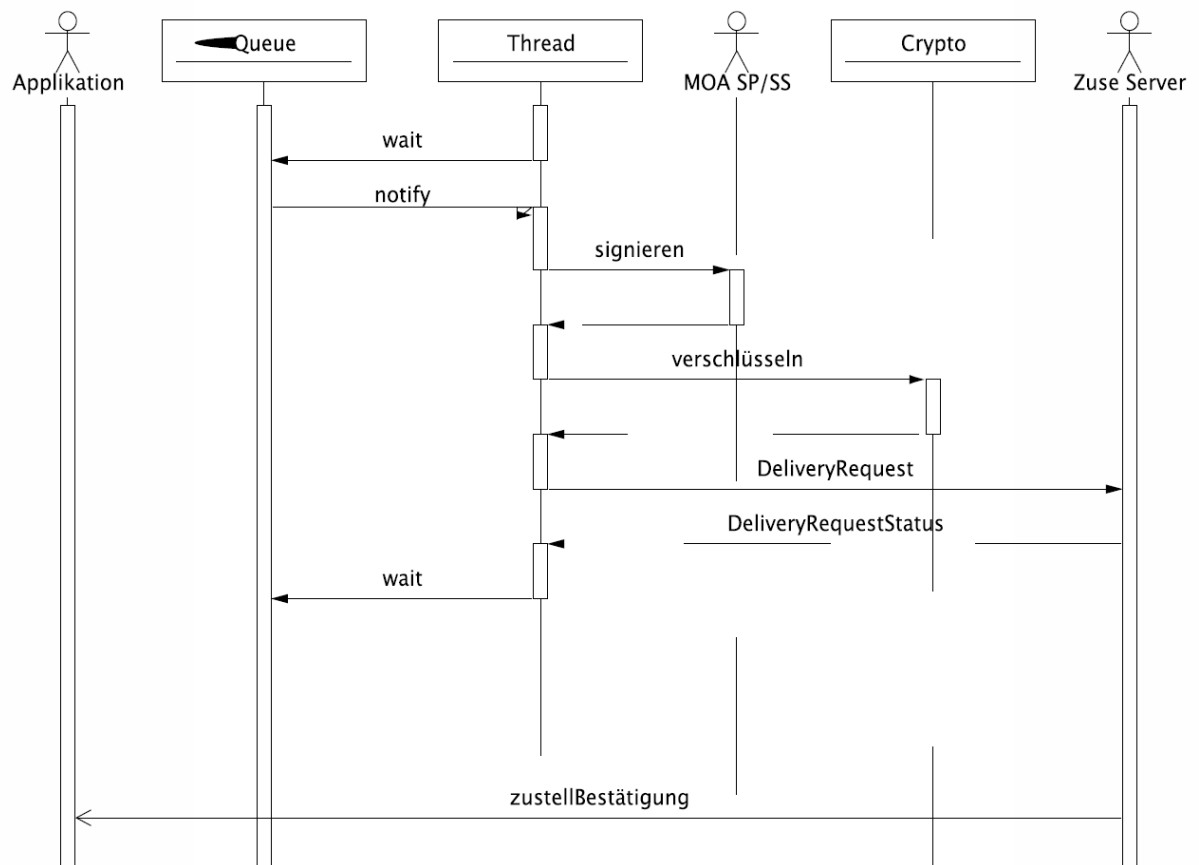
MOA-ZS stellt eine Anfrage an den Zustellkopf, ob der Empfänger elektronische Zustellungen entgegennehmen kann und wählt aus der Liste der zurückgegebenen Zustellserver einen aus. Dabei werden bevorzugt die Zustellserver ausgewählt, bei denen der Empfänger ein X.509 Zertifikat zum Verschlüsseln der Zustellstücke hinterlegt hat. Falls mehrere Zustellserver ein Zertifikat gespeichert haben, wird einer ausgewählt, bei dem der Empfänger keine Einschränkungen hinsichtlich des MIME-Types eingestellt hat. Falls dann immer noch mehrere Zustellserver in der Liste sind, wird, falls eine Liste von bevorzugten Zustellservern in MOA-ZS konfiguriert wurde, einer aus dieser Prioritätsliste ausgewählt. Andernfalls wird einer per Zufall ausgewählt.

Sind alle notwendigen Daten vorhanden und kann an den Empfänger elektronisch zugestellt werden, wird der Request in einer Datenbank persistens gespeichert. Ist MOA-ZS als Cluster installiert bzw. konfiguriert worden, wird durch Ermitteln der Anzahl der Request-Objekte mit der jeweiligen `queue_id` die Queue die am wenigsten ausgelastet ist ausgewählt. Wenn das die eigene Queue ist, wird die `ID` des Zustellstücks aus der Datenbank in die Queue eingetragen. Falls die eigene Queue voll ist oder es eine weniger befüllte Queue gibt, wird das Zustellstück mit der `queue_id - 1` gespeichert, damit es vom nächsten Watchdog-Thread importiert wird (Siehe auch Kapitel 2.1.4 Watchdog-Thread).

War eines der Zustellstücke vom Typ `DocumentReference`, gibt MOA-ZS eine Antwort vom Typ `PartialSuccess` an die Absenderapplikation zurück, ansonsten eine Antwort vom Typ `Success` (Siehe auch Kapitel 2.2 Schnittstellen)

## 1.2 Asynchroner Teil

Abbildung 2, „Asynchroner Teil der Requestverarbeitung“ beschreibt den Ablauf im asynchronen Teil von MOA-ZS.



**Abbildung 2: Asynchroner Teil der Requestverarbeitung**

Abbildung 3, „Asynchroner Teil der Requestverarbeitung (Zustellstück mit Callback-URI)“ beschreibt den Ablauf im asynchronen Teil von MOA-ZS wenn eines der mitgeschickten Dokumente durch eine Callback-URI referenziert wird.

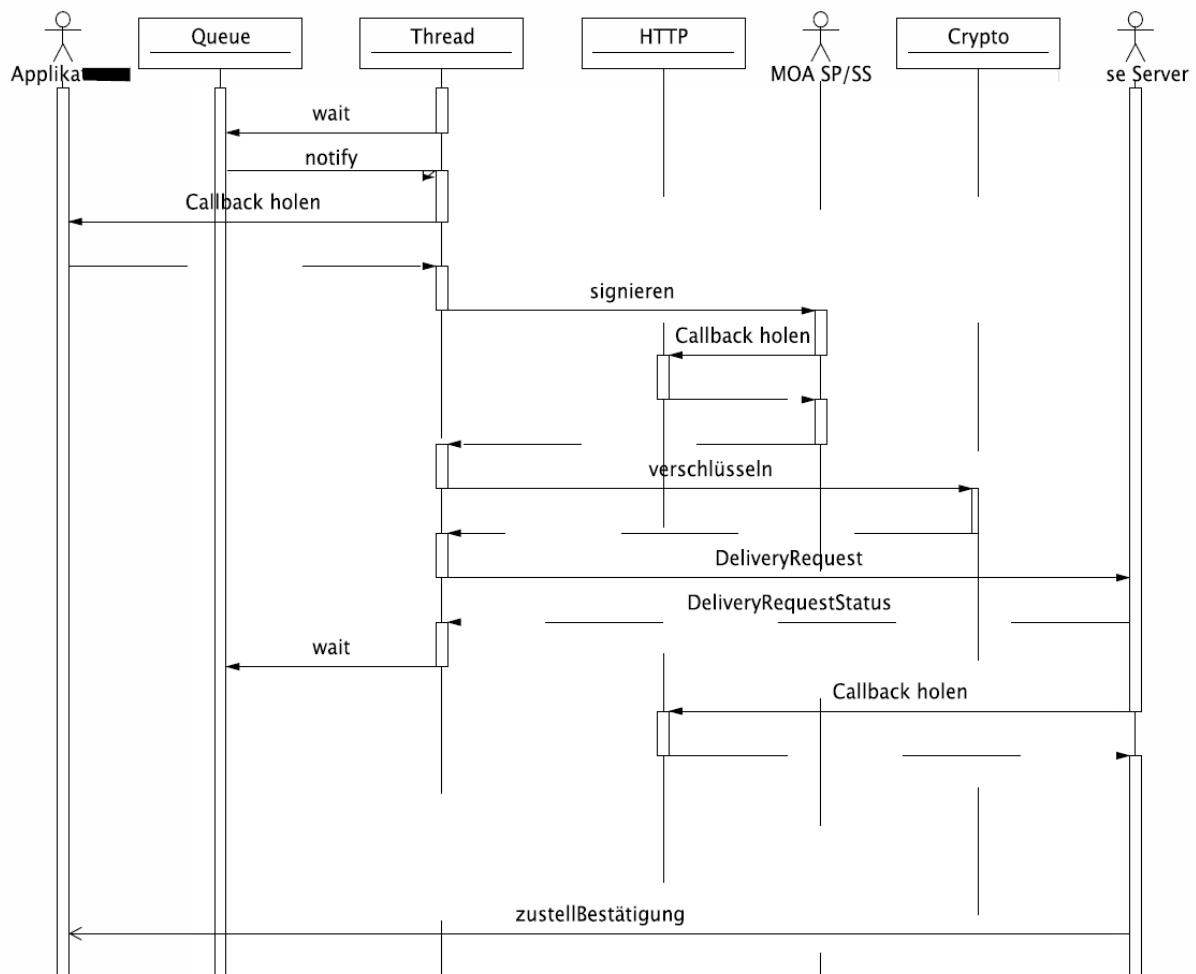


Abbildung 3: Asynchroner Teil der Requestverarbeitung (Zustellstück mit Callback-URI)

## 1.2.1 Aufbereiten des Zustellstücks

Im ersten Schritt wird für Zustellstücke ohne XML-Dokument ein XML-Deckblatt erstellt. Dieses enthält die Absender- und Empfängerdaten und eine Liste der angehängten Dokumente.

Sind Payload-Elemente vom Typ `DocumentReference` vorhanden, wird versucht, die referenzierten Dokumente via HTTP bzw HTTPs nachzuladen. In der Konfigurationsdatei `mo-azs_config.xml` kann eingestellt werden, wie oft und mit welchem zeitlichen Abstand das Nachladen versucht werden soll, falls ein Fehler beim Verbindungsaufbau auftritt.

XML-Dokumente mit einer `XMLProfilID` werden mit den Metadaten, die im Profil eingetragen sind, angereichert.

## 1.2.2 Signieren des Zustellstücks

Im nächsten Schritt wird das Zustellstück mit Hilfe von MOA-SS mit einer Amtssignatur versehen. Dieser Schritt wird (wieder konfigurierbar wie oft) wiederholt, falls keine Verbindung zu MOA-SS aufgebaut werden kann. Falls eine Fehlermeldung von MOA-SS retourniert wird, findet keine weitere Wiederholung statt.

## 1.2.3 Erstellung eines S/MIME Containers

Im nächsten Schritt werden die Dokumententeile zu einem MIME-Container oder Zuse-Container zusammengefasst. Welches Format erstellt wird, kann vom Administrator in der Konfiguration eingestellt werden. Falls ein Zertifikat im Zustellserver eingetragen worden ist,

133 wird dieser Container verschlüsselt (RSA, DES\_EDE). Das verschlüsselte Zustellstück wird als  
134 S/MIME-Container gespeichert.

#### 135 **1.2.4 Übergabe an den Zustellserver**

136 Anschließend wird das Zustellstück an den Zustellserver übergeben. Auch hier können wie-  
137 der eine maximale Anzahl von Versuchen und eine Zeitspanne zwischen den Versuchen  
138 angegeben werden.

#### 139 **1.2.5 Benachrichtigung des Absenders**

140 Im Fall der erfolgreichen Übergabe an einen Zustellserver oder im Fall eines Fehlers, der  
141 nicht durch den Ausfall einer Internetverbindung verursacht wurde, wird an den Absender  
142 eine Benachrichtigung verschickt (SOAP-Request oder Email). Außerdem wird die Meldung  
143 im Logfile verzeichnet und das Requestobjekt inklusive aller Zustellstücke gelöscht.

## 2. Detaillierte Beschreibung von MOA-ZS

### 2.1 Funktionen

#### 2.1.1 Vollständigkeitsprüfung

Da es sich bei den SOAP-Nachrichten, die mit MOA-ZS verarbeitet werden, nicht um RPC Nachrichten, sondern um Nachrichten vom Typ *document/literal* handelt, kommen keine mit WSDL2JAVA generierten Klassen zum Einsatz, sondern der Body der Nachrichten wird als DOM-Objekt an die Funktion übergeben.

Die Prüfung der Pflichtfelder erfolgt mit Hilfe von XPath-Ausdrücken. Dabei werden die statischen Methoden der Klasse `moazs.util.MoaXPath` verwendet. Genauere Details sind in [5] bzw im Quellcode der Klassen enthalten.

Mit deren Methode `getFirstString(Element in, String xpath)` wird der erste Knoten der Ergebnisliste des XPath-Ausdrucks in einen String konvertiert und dann zurückgegeben.

```
MoaXPath.getFirstString(physicalPerson, "p:Name/p:GivenName/text()")
```

Mit diesem Ausdruck wird der Wert des ersten Textknotens unterhalb von `p:GivenName` ermittelt.

Mit der Methode `getFirstElement(Element in, String xpath)` wird der erste Knoten der Ergebnisliste des XPath-Ausdrucks als `Element` zurückgegeben. Dieses Element kann dann als Startpunkt für weitere XPath Anfragen verwendet werden.

Mit der Methode `getAllElement(Element in, String xpath)` wird die gesamte Ergebnisliste des XPath-Ausdrucks zurückgegeben. Diese Methode kommt vor allem zum Einsatz, um über Blöcke zu iterieren, die in einem Request mehrmals vorkommen können (z.B.: Adressen, Binärdateien, usw.)

#### 2.1.2 Lastverteilung

Wenn die Anzahl der Zustellstücke in der Queue einen konfigurierbaren Schwellwert übersteigt, startet die Queue einen neuen Workerthread (bis zu einem konfigurierbaren Maximalwert).

Die Workerthreads bekommen die Request-ID in der `doSomething`-Methode übergeben und laden das zugehörige Requestobjekt aus der Datenbank. Anhand des Status des Request-Objekts wird eine der Verarbeitungsmethoden ausgewählt.

#### 2.1.3 Auswahl des Zustellservers

Wenn eine Zusekopf-Anfrage mehrere Zustellserver findet, wird zunächst eine Liste aller Zustellserver erstellt, bei denen der Empfänger ein Zertifikat zur Verschlüsselung der Zustellstücke installiert hat. Falls keiner der Zustellserver ein Zertifikat bereithält oder mehrere Zustellserver ein Zertifikat installiert haben, werden aus den Verbleibenden diejenigen gefiltert, die als MimeType `"*/"` angegeben haben. Sollten dann immer noch mehr als einer überbleiben oder keiner der Zustellserver den Mimetype `"*/"` akzeptieren, wird jener ausgewählt, welcher in der Konfiguration als bevorzugter eingetragen wurde. Andernfalls wird einer per Zufall ausgewählt (siehe Zustellgesetz).

Die Auswahl der Zustellserver erfolgt im synchronen Teil eines Request. Dabei verwendet die Methode `getZuseserver` der Klasse `moazs.api.MoaSync` die Methoden der Klasse `moazs.external.ZuseRequest`. Genauere Details sind in [5] bzw im Quellcode der Klassen enthalten.

## 2.1.4 Watchdog-Thread

Der Watchdog-Thread schreibt periodisch einen Timestamp in die Tabelle *queue\_info* und prüft dann, ob es Einträge anderer Clusterknoten gibt, deren Timestamp eine gewisse Zeit lang nicht aktualisiert worden sind. Falls eine solche Queue gefunden wird, wird versucht, einen HTTP-Request an die in der Tabelle angegebenen "Ping"-URL abzusetzen. Wenn dieser Request nicht beantwortet wird, werden alle Einträge in der Request-Tabelle mit der entsprechenden Queue-id auf -1 gesetzt.

Im nächsten Durchlauf sammelt der Watchdog-Thread alle Einträge in der Request-Tabelle ein, die die *queue\_id* -1 haben und importiert sie in die eigene Queue bis diese voll ist. In einem Cluster versuchen das möglicherweise mehrere Knoten gleichzeitig, der Zugriff ist aber transaktionsgeschützt, sodaß nur der erste Knoten Erfolg hat. Der Watchdog-Thread wird von einem *ServletContextListener* beim Start von MOA-ZS gestartet. Genauere Informationen können in [5] bzw dem Quellcode der Klasse *moazs.watchdog.Watchdog* entnommen werden.

## 2.1.5 Signaturerstellung

MOA-ZS signiert optional die Zustellstücke mit Hilfe von MOA-SS. Dabei wird eine XML-Signatur in das XML-Dokument des Zustellstücks bzw in das XML-Deckblatt des Zustellstücks eingebunden. In welchen Knoten die Signatur eingefügt werden soll, kann der Absender mit Hilfe des Signatur-XPath-Ausdrucks im Request festlegen. Die Signatur wird im enveloped-Format erstellt, für alle zusätzlichen Dokumente, die verschickt werden, wird ein *DataObjectInfo* Block angelegt, der das Dokument und eine Referenz auf den lokalen Dateinamen beim Empfänger enthält. Mit diesen Daten erstellt MOA-SS eine Detached-Signatur. Dabei speichert der Signatur-Knoten die HashWerte der Dokumente und das Ergebnis der Signaturberechnung.

Clientseitig ist es dadurch möglich, die Signatur des XML-Dokuments unabhängig von der Signatur der mitgeschickten Dokumente zu überprüfen, d.h. es kann die Korrektheit der Signatur auch bei Fehlen eines Dokumentes für den erhaltenen Teil überprüft werden.

## 2.1.6 Verschlüsselung

Vor dem Verschlüsseln wird das Dokument in einen MIME-Container oder einen Zusecontainer verpackt. Das Format, das erstellt werden soll, kann mit Hilfe der Konfigurationsdatei *moazs\_config.xml* ausgewählt werden. Das resultierende Objekt wird dann mit dem Zertifikat des Users verschlüsselt und in einen SMIME-Container verpackt.

Dabei kommt gemäß der Spezifikation des SMIME Standards als symetrischer Verschlüsselungsalgorithmus Triple-DES im CBC-Modus zum Einsatz. Das ist ein Standard, den alle SMIME-Clients unterstützen müssen und der verwendet werden soll, wenn dem Absender nicht bekannt ist, ob der Empfänger eine stärkere Verschlüsselung akzeptiert.

## 2.2 Schnittstellen

### 2.2.1 Applikation an MOA-ZS

Die SOAP-Requests, über die die Absenderapplikation mit MOA-ZS kommuniziert, verwenden *DeliveryRequest*, um eine Anfrage zu senden. Die dazugehörige Antwort ist vom Typ *DeliveryResponse*. Der *DeliveryRequest* besteht aus den Elementen *Sender*, *Receiver*, *MetaData*, *XMLDocument* und *Payload*. Dabei ist *XMLDocument* optional und *Payload* kann 0 bis n mal vorkommen.

#### **Sender**

Im `Sender`-Element werden die ID des Absenderprofils und der Name der Schlüsselgruppe aus MOA-SS übertragen. Falls das Dokument nicht signiert werden soll, muss der Name der Schlüsselgruppe leer sein.

```
<Sender>
  <ProfileID>absender123</ProfileID>
  <SignatureProfileID>SchluesselgruppenName</SignatureProfileID>
</Sender>
```

## Receiver

Im `Receiver` Element sind die Empfängerinformationen enthalten. Das MOA-ZS Schema verwendet dabei Elemente aus einer simplifizierten Version des `PersonData 2.0` Schemas. Ein Empfänger kann dabei auf mehrere Arten adressiert werden.

- unverschlüsselte Zustell-bPK des Empfängers
- verschlüsselte Zustell-bPK des Empfängers
- bPK des Verfahrensbereichs der Absenderapplikation
- Name und Verständigungsadresse (und Geburtsdatum bei RSa-Zustellung)
- Name und Postadresse (und Geburtsdatum bei RSa-Zustellung)
- Stammzahl von nicht natürlichen Personen (z.B. Firmenbuchnummer, Vereinsnummer, usw.)

In jedem Fall muß ein `Person`-Element vom Typ `p:PhysicalPerson` oder vom Typ `p:CorporateBody` vorhanden sein. Bei einer natürlichen Person muß ein Feld vom Typ `p:Name` vorhanden sein, bei einer RSa-Zustellung auch ein Feld `p:DateOfBirth`.

```
<Receiver>
  <p:PhysicalPerson>
    <p:Name>
      <p:GivenName>Max</p:GivenName>
      <p:FamilyName>Mustermann</p:FamilyName>
    </p:Name>
    <p:DateOfBirth>1975-01-15</p:DateOfBirth>
  </p:PhysicalPerson>
</Receiver>
```

Für eine nichtnatürliche Person werden die Elemente `p:Fullname` und `p:Organisation` (optional) verwendet.

```
<Receiver>
  <p:CorporateBody>
    <p:Fullname>FooBar GmbH<p:Fullname>
    <p:Organisation>RnD Abteilung</p:Organisation>
  </p:CorporateBody>
</Receiver>
```

274

275 Wenn keine bPK zur Identifikation verwendet wird, muß zusätzlich zu den Personen-  
276 daten mindestens eine Adresse angegeben werden. Adressen können mehrmals  
277 vorkommen und werden in einem Address-Element übergeben, welches vom Typ  
278 p:PostalAddress, p:InternetAddress oder p:TelephoneAddress sein kann.

279

280

```
<Receiver>
```

281

```
<p:PhysicalPerson>
```

282

```
<p:Identification>
```

283

```
<p:Value>A123124123ES...</p:Value>
```

284

```
<p:Type>urn:publicid:gv.at:cdid+ZS</p:Type>
```

285

```
</p:Identification>
```

286

```
</p:PhysicalPerson>
```

287

```
<p:InternetAddress>
```

288

```
<p:Address>roadrunner@acme.com</p:Address>
```

289

```
</p:InternetAddress>
```

290

```
</Receiver>
```

291

292 Eine Adresse vom Typ p:InternetAddress hat genau ein Unterelement vom Typ  
293 p:Address, in dem die Emailadresse enthalten ist.

294 Eine Adresse vom Typ p:TelephonAddress enthält ein Unterelement vom Typ  
295 p:Number, in dem die Telefonnummer enthalten ist.

296 Eine Adresse vom Typ p:PostalAddress enthält die Elemente p:CountryCode  
297 (Ländercode optional), p:PostalCode (PLZ), p:Municipality (Ort) und  
298 p:DeliveryAddress. Letzteres enthält dabei die Unterelemente p:StreetName,  
299 p:BuildingNumber, p:Unit (Stiege optional) und p:DoorNumber (optional)

300

301

```
<p:PostalAddress>
```

302

```
<p:CountryCode>AT</p:CountryCode>
```

303

```
<p:PostalCode>1234<p:PostalCode>
```

304

```
<p:Municipality>Sonstwo</p:Municipality>
```

305

```
<p:DeliveryAddress>
```

306

```
<p:StreetName>Foostraße<p:StreetName>
```

307

```
<p:BuildingNumber>42</p:BuildingNumber>
```

308

```
<p:Unit>4</p:Unit>
```

309

```
<p:DoorNumber>2</p:DoorNumber>
```

310

```
</p:DeliveryAddress>
```

311

```
</p:PostalAddress>
```

312

### 313 **MetaData**

314 Das MetaData-Element enthält die Elemente AppDeliveryID (eine ID, unter der das  
315 Zustellstück von der Absenderapplikation gespeichert ist), DeliveryQuality (enthält  
316 entweder den String RSa oder nonRSa) und das Element RequiresEncryption  
317 (vom Typ boolean).

318

319

```
<MetaData>
```

```

320         <AppDeliveryID>DOK1234</AppDeliveryID>
321         <DeliveryQuality>non-RSa</DeliveryQuality>
322         <RequiresEncryption>true</RequiresEncryption>
323     </Metadata>
324

```

## 325 XMLDocument

326 Ein `XMLDocument` Element enthält das Unterelement `XMLContent` vom Typ `xs:any`.  
 327 In diesem Element kann ein zu verschickendes XML-Dokument eingebettet werden.  
 328 Weiters wird entweder eine `XMLProfileID` angegeben oder ein `FileName`, `MIMETy-`  
 329 `pe`, `ResultingMIMEType`, `SignatureXPath`, `SignatureStylesheet` und ein `Pre-`  
 330 `viewStylesheet` Element. Dabei ist im `FileName`-Element der Dateiname unter  
 331 dem das Dokument beim Empfänger gespeichert wird, enthalten. Im `MIMEType` Ele-  
 332 ment wird der `Mimetype` des XML-Dokuments gespeichert, im Element `Resulting-`  
 333 `MIMEType` der `Mimety` nach Anwendung des `PreviewStylesheets` für die Bürgerkar-  
 334 tenumgebung des Empfängers. Mit Hilfe des Elements `SignatureXPath` und seinem  
 335 Attribut `Index` kann spezifiziert werden, unter welchem Knoten die `XMLDSIG-`  
 336 `Signatur` von MOA-SS in das XML-Dokument eingefügt werden soll. Die Elemente  
 337 `SignatureStylesheet` und `PreviewStylesheet` enthalten jeweils ein Unterele-  
 338 ment `XMLContent`, das ebenfalls vom Typ `xs:any` ist und ein `XSLT-Stylesheet` ent-  
 339 halten soll. Das `Signatur-Stylesheet` wird vor der Berechnung des Hashwerts für die  
 340 `Signaturerstellung` durch MOA-SS angewandt und ist optional. Das `PreviewStyles-`  
 341 `heet` enthält außerdem ein `FileName` Element, das wie auch beim XML-Dokument  
 342 den Dateinamen beim Empfänger angibt.

```

343
344 <XMLDocument>
345     <XMLContent>
346         <foo:foodoc xmlns:foo="urn:foons">ich bin ein xmldokument</foo:foodoc>
347     </XMLContent>
348     <FileName>foo.xml</FileName>
349     <MIMEType>application/x-foo</MIMEType>
350     <ResultingMIMEType>text/xhtml</ResultingMIMEType>
351     <SignatureXPath Index="0">/foo:foodoc</SignatureXPath>
352     <PreviewStylesheet>
353         <XMLContent>
354             <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSLT"
355             xmlns:foo="urn:foons">
356                 <xsl:template match='foo:foobar'>
357                     <html><body><xsl:value-of select="."/></body></html>
358                 </xsl:tempalte>
359             </xsl:stylesheet>
360         </XMLContent>
361         <FileName>preview.xsl</FileName>
362     </PreviewStylesheet>
363 </XMLDocument>
364

```

## 365 Payload

366 Ein `Payload` Element enthält entweder ein `BinaryDocument` oder ein `DocumentReference`  
 367 Element. Im `BinaryDocument` Element werden die Dokumente Base64-codiert übergeben.

Dabei kommen die Elemente `Base64Content`, `FileName` und `MIMEType` zum Einsatz. `DocumentReference` enthält die Unterelemente `URL`, `FileName`, `MIMEType` und optional `MD5Checksum`. Das Element `URL` enthält die URL unter der MOA-ZS das Dokument via HTTP oder HTTPS abholen kann.

```
<Payload>
  <BinaryDocument>
    <Base64Content>ASIDOJAISDJ2340IAJD2DDD...<Base64Content>
    <FileName>bescheid.pdf</FileName>
    <MIMEType>application/pdf<MIMEType>
  </BinaryDocument>
</Payload>

<Payload>
  <DocumentReference>
    <URI>http://server/bescheid.pdf</URI>
    <FileName>bescheid.pdf</FileName>
    <MIMEType>application/pdf<MIMEType>
    <MD5Checksum>88ffaa8a88888a8a88a883</MD5Checksum>
  </DocumentReference>
</Payload>
```

Dokumente können an MOA-ZS auch mit "Soap with Attachments" (SwA) übergeben werden. In diesem Fall kommt kein `Payload`-Element zum Einsatz, da alle benötigten Daten (Mimetype, Dateiname, ...) als MIME-Header im SwA-Part mitgeschickt werden.

## 2.2.2 MOA-ZS an Applikation

Antworten des synchronen Teils an die Applikation erfolgen im `DeliveryResponse`. Dieses Element enthält je nach Antwort ein `Success`, `Error` oder `PartialSuccess` Element. `PartialSuccess` wird verschickt, wenn die Applikation ein Dokument mit Callback-URI verschickt hat. Alle drei Elemente haben ein `AppDeliveryID` und `MZSDeliveryID` Element. `MZSDeliveryID` ist die ID des Zustellstücks in MOA-ZS, `AppDeliveryID` die ID des Zustellstücks der Absenderapplikation. Im Fehlerfall kommen auch noch das Element `Code` und `Text`. Dort werden eine Fehlernummer und eine textuelle Beschreibung eingetragen.

Wenn im asynchronen Teil eine Meldung an die Applikation verschickt werden muß, kommt eine Nachricht vom Typ `DeliveryNotification` zum Einsatz, deren Aufbau ähnlich ist wie der von `DeliveryResponse`. Im Fall einer Meldung über die Übergabe des Zustellstücks an einen Zustellserver enthält es zusätzlich das Element `DeliveryStatement` mit den Unterelementen `DeliveryServer`, `Timestamp` und `ZSDeliveryID`.

## 2.2.3 Zusekopf

Die Schnittstelle des *Zusekopf* ist im Detail in [3] beschrieben.

## 2.2.4 Stammzahlenregister

Die Schnittstelle zum SZR ist im Detail in [2] im Kapitel *Prüfung der Adressierbarkeit* beschrieben.

## 2.2.5 MOA-SS

Die Schnittstelle zu MOA-SS ist im Detail in [2] im Kapitel „Anbringen der Absendersignatur“ beschrieben.

## 2.2.6 Zustellserver

Die Schnittstelle des *Zustellserver* ist im detail in [4] beschrieben.

## 2.3 Datenorganisation

MOA-ZS speichert alle Requestdaten in einer relationalen Datenbank. Die Daten werden allerdings nur solange in der Datenbank gespeichert, bis der Request abgearbeitet ist, dann werden sie gelöscht. Die Datenbank wird also verwendet, um Transaktionsicherheit und einen geordneten Wiederanlauf nach einem Systemausfall zu gewährleisten. Abbildung 4, „Datenmodell MOA-ZS“ zeigt das Datenmodell als UML-Diagramm.

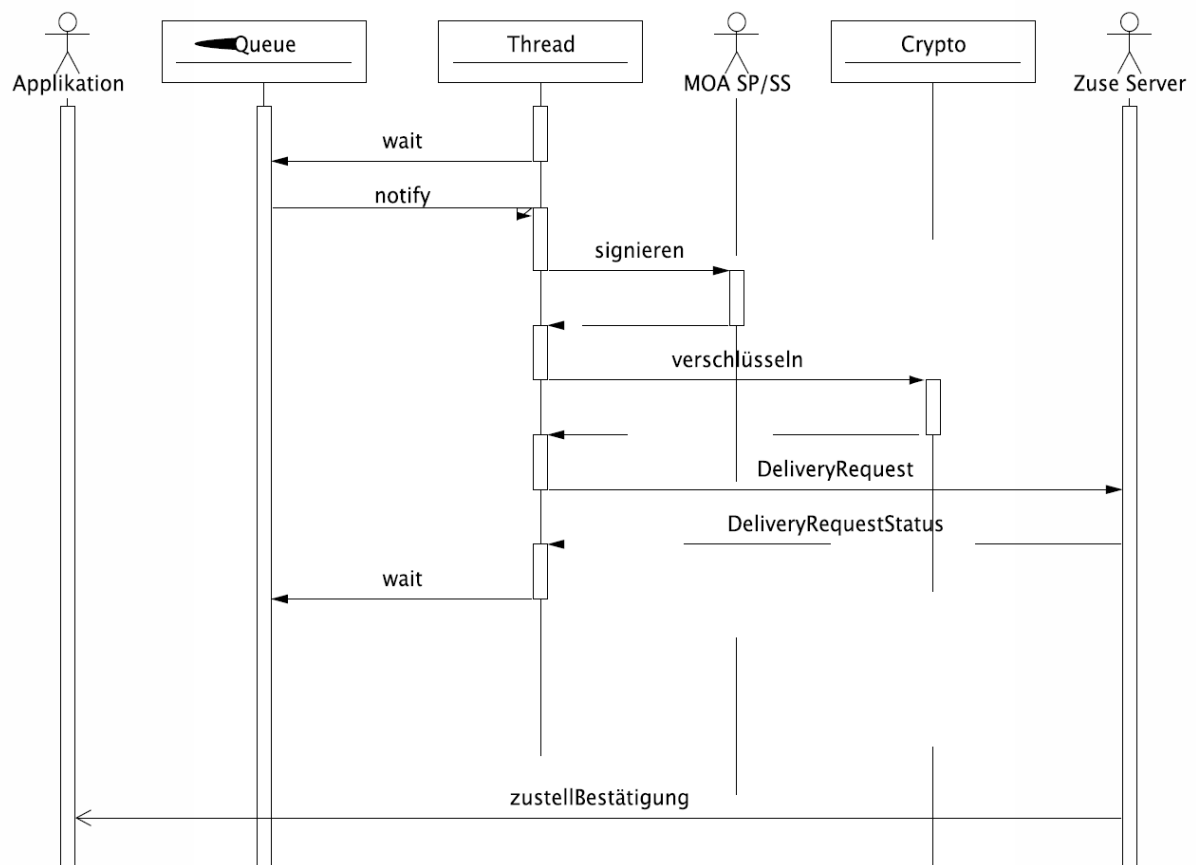


Abbildung 4: Datenmodell MOA-ZS

In der Tabelle "request" werden die Metadaten, die Absenderdaten und der aktuelle Status in dem sich das Zustellstück befindet, gespeichert. Falls bei einem Ausfall eines Backendsystems eine Wiederholung der Aktion notwendig ist werden außerdem die Anzahl der bisherigen Versuche und der früheste Zeitpunkt des nächsten Versuchs gespeichert.

### 2.3.1 Tabelle payload

In der Tabelle "payload" werden alle Dokumente, die mit einem Zustellstück mitgeliefert werden, abgelegt. Der eigentliche Inhalt des Dokuments wird dabei in einem Feld vom Typ `Blob` oder `Bytea` abgelegt (je nach Datenbank). Das Feld `base64` zeigt an, ob der Inhalt des Blobs derzeit im base64 Format gespeichert ist oder nicht.

## 2.3.2 Tabelle xmldocument

In der Tabelle "*xmldocument*" werden die Daten des XML-Dokuments oder des XML-Deckblatts gespeichert. Der eigentliche Inhalt bzw die Stylesheets wird wie bei der payload-Tabelle in einem `Blob`- bzw `Bytea`-Feld abgelegt.

## 2.3.3 Tabelle rcpt

Die Tabelle "*rcpt*" enthält alle Daten zu einem Empfänger. Im Feld *zusebpk* wird die (verschlüsselte) Zustell-bPK gespeichert.

## 2.3.4 Tabelle identification

In der Tabelle "*identification*" werden eventuell vorhandene weitere bPK eines Empfängers gespeichert.

## 2.3.5 Tabelle internetaddress

In der Tabelle "*internetaddress*" werden die Emailadressen des Empfängers gespeichert.

## 2.3.6 Tabelle telephone

In der Tabelle "*telephone*" werden die Telefonnummern des Empfängers gespeichert.

## 2.3.7 Tabelle postaladdress

In der Tabelle "*postaladdress*" werden die Adressen des Empfängers gespeichert.

## 2.3.8 Tabelle Status

In der Tabelle "*Status*" werden die möglichen Stati, in denen sich ein Zustellstück befinden kann, gespeichert.

## 2.3.9 Tabelle delivery\_quality

In der Tabelle "*delivery\_quality*" werden die möglichen Werte des Felds *delivery\_quality* gespeichert.

## 2.3.10 Tabelle zuseserver

In der Tabelle "*zuseserver*" wird das Ergebnis der Zustellkopf-Anfrage gespeichert.

## 2.3.11 Tabelle queue\_info

In der Tabelle "*queue\_info*" werden die Timestamps des Watchdogthreads gespeichert.

### 3. Referenzen

[1]	MOA-ZS Administrationshandbuch
[2]	MOA-ZS Technische Dokumentation
[3]	Elektronische Zustellung – Zustellkopf – Schnittstellenspezifikation 1.3.0
[4]	Elektronische Zustellung – Message Spezifikation
[5]	MOA-ZS Javadoc – docs/apidocs/index.html
[6]	Bouncycastle JCE Provider – <a href="http://www.bouncycastle.org">http://www.bouncycastle.org</a>
[7]	Apache AXIS - <a href="http://ws.apache.org/axis/">http://ws.apache.org/axis/</a>
[8]	Jakarta HTTP Components - <a href="http://hc.apache.org">http://hc.apache.org</a>
[9]	JConfig – <a href="http://www.jconfig.org">http://www.jconfig.org</a>

## 4. Historie

Version 0.1	Datum 01.11.2004	Kommentar Erstellung
Ersteller Nikolaus Gradwohl		
Version 0.11	Datum 04.11.2004	Kommentar Revision
Ersteller Andreas Erlacher		
Version 0.2	Datum 05.11.2004	Kommentar Einarbeitung Revisionssergebnisse
Ersteller Nikolaus Gradwohl		
Version 0.21	Datum 07.11.2004	Kommentar Revision
Ersteller Andreas Erlacher		
Version 0.3	Datum 10.11.2004	Kommentar Einarbeitung Revisionssergebnisse
Ersteller Nikolaus Gradwohl		
Version 0.31	Datum 10.11.2004	Kommentar Revision
Ersteller Andreas Erlacher		
Version 0.4	Datum 10.11.2004	Kommentar Einarbeitung Revisionssergebnisse
Ersteller Nikolaus Gradwohl		
Version 1.1.0	Datum 07.11.2008	Kommentar Anpassungen an die MOA-ZS Version 1.1.0
Ersteller Arne Tauber		

463

## 5. Anhang

464

### 5.1 Begriffe und Abkürzungen

Begriff, Abkürzung	Beschreibung
MOA-ZS	MOA Zustellservice
MOA-SS/SP	MOA Signaturservice/Signaturprüfung
DOM	Document Object Model
bPK	Bereichsspezifisches Personenkennzeichen
zbPK	Bereichsspezifisches Personenkennzeichen für den Bereich Zustellung

465

Tabelle 5-1 Begriffe und Abürzungen

466

### 5.2 Tabellenverzeichnis

467

468

Tabelle 9-1 Begriffe und Abürzungen.....	19
--	----

469

### 5.3 Abbildungsverzeichnis

470

471

472

473

474

475

Abbildung 1: Synchroner Teil der Requestverarbeitung .....	4
Abbildung 2: Asynchroner Teil der Requestverarbeitung .....	6
Abbildung 3: Asynchroner Teil der Requestverarbeitung (Zustellstück mit Callback-URI) .....	7
Abbildung 4: Datenmodell MOA-ZS .....	15

## 476 **5.4 Konfigurationsdatei moazs\_config.xml**

477 TBD - Hier beispielhafte Konfigurationsdatei eintragen

478